Desenvolvimento de uma Mesa Virtual Híbrida para RPG com Projeção e Rastreamento por Sensor de Profundidade

Arthur de Souza Spironello
Curso de Ciências de Computação
Universidade Franciscana
Santa Maria, Brasil
arthur.spironello@ufn.edu.br

Alessandro André Mainardi de Oliveira
Curso de Ciências de Computação
Universidade Franciscana
Santa Maria, Brasil
amainardi@ufn.edu.br

***Resumo*—Este trabalho apresenta o desenvolvimento de um protótipo de *Virtual Tabletop* (VTT) em Unity/C#, voltado ao apoio de sessões de RPG de mesa em ambientes digitais e híbridos. O sistema foi projetado de forma modular, contemplando funcionalidades como carregamento de mapas, organização em camadas, criação e persistência de personagens, rolagem de dados, névoa de guerra e separação entre a visualização do mestre e dos jogadores. Além do uso em modo digital, o projeto inclui suporte a uma mesa física interativa, utilizando projeção e rastreamento de peças por sensor de profundidade para aproximar a automação dos VTTs da tangibilidade dos jogos presenciais. Dessa forma, o trabalho busca reduzir a distância entre ferramentas digitais de apoio ao RPG e a experiência social e física característica das partidas tradicionais.**

***Palavras-chave*: Jogos Digitais, RPG, *Virtual Tabletop*, Unity, Interação Humano-Computador, Realidade Aumentada Projetiva, Kinect.**

I. Introdução

Os jogos de tabuleiro constituem uma forma de entretenimento social baseada na interação entre participantes em um espaço compartilhado. Entre seus diferentes gêneros, o *Role-Playing Game* (RPG) de mesa destaca-se pela construção colaborativa de narrativas, na qual jogadores interpretam personagens em um universo ficcional mediado por regras e pela atuação de um mestre de Jogo. Diferentemente de jogos com objetivos previamente definidos, o RPG valoriza a criatividade, a improvisação e a interação social como elementos centrais da experiência [1].

Apesar de sua riqueza narrativa, uma sessão de RPG pode envolver diversos elementos de controle, como fichas de personagens, mapas, dados, miniaturas, atributos, estados e regras específicas de cada sistema. Esse conjunto de informações pode aumentar a carga de gestão da sessão, especialmente em campanhas longas ou sistemas com maior complexidade mecânica. Nesse contexto, os *Virtual Tabletops* (VTTs) surgem como ferramentas digitais de apoio ao RPG, permitindo organizar mapas, personagens, rolagens de dados e elementos visuais em uma interface computacional [2]. Tais plataformas contribuem para agilizar a preparação e condução das partidas, além de possibilitar recursos visuais como mapas interativos e névoa de guerra [3].

Entretanto, a adoção de soluções totalmente digitais pode reduzir aspectos importantes da experiência presencial, como a manipulação física de peças, a atenção compartilhada sobre a mesa e a copresença entre os jogadores. A mediação por telas individuais tende a deslocar parte da interação para a interface, diminuindo a centralidade do espaço físico da partida [4]. Por esse motivo, abordagens híbridas, que combinam elementos digitais e físicos, tornam-se relevantes para preservar a tangibilidade dos jogos de mesa ao mesmo tempo em que incorporam recursos de automação e visualização.

A Realidade Aumentada, em especial em configurações com projeção sobre superfícies físicas, apresenta-se como alternativa para esse cenário ao permitir que informações digitais sejam visualizadas diretamente no ambiente compartilhado [5]. Trabalhos recentes em mesas interativas por projeção e objetos tangíveis indicam ganhos de naturalidade e redução de esforço de interação quando comparados a abordagens fortemente mediadas por dispositivos individuais [6]. Assim, este trabalho propõe a integração de recursos de automação e visualização típicos de VTTs a uma experiência híbrida de RPG, mantendo a interação física compartilhada em torno da mesa e preservando a dinâmica presencial da partida.

Dessa forma, o objetivo do trabalho é projetar e implementar um protótipo funcional de VTT modular em Unity/C#, concebido como aplicação *desktop* e estruturado para suportar dois modos de uso: (i) operação digital em monitor e (ii) extensão para operação híbrida por projeção e rastreamento físico. O protótipo contempla funcionalidades essenciais de apoio a sessões de RPG, incluindo carregamento de mapas, organização em múltiplos tabuleiros (camadas), criação e gerenciamento de personagens, manipulação de *tokens* — representações digitais de personagens, criaturas ou objetos no mapa —, rolagem de dados, controle de névoa de guerra e visualização separada para mestre e jogadores. A integração com sensor de profundidade é tratada como extensão do sistema, explorando a capacidade desse tipo de dispositivo de capturar informações de profundidade do ambiente [7] e inspirando-se em sistemas

tabletop com rastreamento físico e projeção [6], [20].

Para viabilizar essa integração, o trabalho considera também desafios técnicos de rastreamento por profundidade, calibração entre espaço físico e digital, latência de atualização e sincronização entre peças reais e tokens virtuais.

Este documento está organizado da seguinte forma: a Seção II apresenta o referencial teórico, a Seção III discute os trabalhos correlatos, a Seção IV descreve a metodologia e o desenvolvimento do protótipo, a Seção V apresenta os resultados obtidos e a Seção VI expõe as conclusões, limitações e trabalhos futuros.

II. Referencial Teórico

Esta seção apresenta os conceitos que fundamentam o desenvolvimento do protótipo proposto, relacionando o contexto dos RPGs de mesa, as plataformas*Virtual Tabletop* (VTT), a Realidade Aumentada Projetiva e as tecnologias utilizadas na implementação. A discussão busca delimitar os elementos teóricos necessários para compreender a proposta de uma mesa virtual híbrida, na qual recursos digitais de apoio à sessão são integrados à interação física e compartilhada característica das partidas presenciais.

A. RPG de Mesa e Virtual Tabletops

O *Role-Playing Game* (RPG) de mesa é um gênero de jogo baseado na construção colaborativa de narrativas, no qual os participantes interpretam personagens em um universo ficcional mediado por regras. Em geral, a dinâmica envolve jogadores, responsáveis por controlar personagens individuais, e um mestre de Jogo, encarregado de descrever o cenário, arbitrar regras e conduzir os desafios da sessão [1]. Diferentemente de jogos com objetivos fixos e lineares, o RPG valoriza a improvisação, a tomada de decisões e a interação social como elementos centrais da experiência. A atividade central do RPG prioriza a contação de histórias (*storytelling*) interativa em vez da vitória. As ações são resolvidas pela integração entre a interpretação dos jogadores e um sistema de regras que utiliza dados para introduzir aleatoriedade [1]. Tal estrutura permite que a narrativa se adapte dinamicamente às escolhas feitas, resultando em uma experiência única a cada sessão.

Durante uma partida, diversos elementos precisam ser gerenciados simultaneamente, como fichas, atributos, estados, mapas, anotações e posicionamento de peças. Nesse contexto, os *Virtual Tabletops* (VTTs) surgem como ferramentas digitais de apoio a jogos de mesa, oferecendo recursos para organizar mapas, personagens, rolagens de dados e elementos visuais em uma interface computacional [2]. Embora essas plataformas facilitem a condução da sessão e viabilizem partidas remotas, sua adoção em formato exclusivamente digital pode deslocar parte da experiência presencial para telas individuais, reduzindo a manipulação física e a atenção compartilhada sobre a mesa [4].

Neste trabalho, o VTT é tratado como a base digital do sistema. O objetivo não é substituir completamente a experiência física, mas oferecer recursos computacionais de apoio à sessão, como mapas, camadas, *tokens*, fichas, rolagens e névoa de guerra, preservando a possibilidade de uso em um ambiente presencial ou híbrido.

B. Realidade Aumentada

A Realidade Aumentada (RA) caracteriza-se pela integração de conteúdo virtual ao ambiente real, de modo que ambos coexistam na percepção do usuário. Um sistema de RA é usualmente descrito por três propriedades: (i) combinar elementos reais e virtuais, (ii) permitir interação em tempo real e (iii) apresentar registro espacial consistente em três dimensões [5].

Diferentes paradigmas de RA podem ser classificados pela forma de visualização e composição do conteúdo virtual. Entre eles destacam-se abordagens *optical see-through*, que sobrepõem gráficos à visão direta por meio de visores semitransparentes, e *video see-through*, que combinam digitalmente imagens capturadas por câmeras antes da exibição ao usuário. Também é comum a implementação em dispositivos portáteis, como smartphones e tablets, em que a composição ocorre na tela do dispositivo [5].

Este trabalho foca na Realidade Aumentada Projetiva, também denominada *Spatial Augmented Reality* (SAR), caracterizada pela projeção de informações digitais diretamente sobre superfícies físicas do ambiente. Nesse paradigma, o conteúdo virtual é compartilhado no espaço comum e pode ser observado simultaneamente por múltiplos usuários sem a necessidade de dispositivos individuais [5]. Em particular, aplicações de projeção aumentada voltadas a jogos e experiências em mesa demonstram a viabilidade de integrar projeção e rastreamento de elementos físicos para ampliar a interação sem descaracterizar a dinâmica presencial [8]. Essa escolha é coerente com a natureza colaborativa do RPG de mesa, no qual a visualização conjunta e a manipulação tátil de peças são elementos centrais da experiência.

C. Tecnologias para Programação

A escolha das tecnologias está relacionada à necessidade de construir uma aplicação interativa, modular e capaz de integrar renderização 2D, entrada do usuário e interoperabilidade com componentes externos. Para isso, foram adotados a *engine* Unity, a linguagem C#, e um módulo nativo de visão computacional e rastreamento (baseado em OpenCV) integrado por interoperabilidade entre código gerenciado e nativo.

1) Unity e C#: Unity é um motor de jogos (*game engine*) multiplataforma utilizada no desenvolvimento de aplicações interativas 2D e 3D. Sua arquitetura é baseada em *GameObjects* e componentes, o que favorece a organização modular de elementos visuais, comportamentos e interfaces [9]. Essa característica foi relevante para o desenvolvimento do protótipo, pois permitiu separar funcionalidades como tabuleiro, *tokens*, mapas, camadas, névoa de guerra e interface do mestre em componentes específicos.

C# é uma linguagem orientada a objetos e com tipagem segura, que organiza programas por meio de tipos, classes, métodos e outros membros [12]. No contexto deste trabalho, C# foi utilizada como linguagem principal dos *scripts* da Unity, implementando a lógica do VTT.

A renderização do tabuleiro utiliza *sprites*, exibidos por meio de componentes como `SpriteRenderer`, enquanto a grade e alguns elementos de *feedback* visual são gerados de forma procedural com `LineRenderer` [10], [11]. Dessa forma, a Unity atua como camada de apresentação e interação do VTT, tanto no modo digital quanto no modo híbrido, recebendo dados processados externamente e atualizando os objetos da cena.

2) C/C++ (módulo nativo): C e C++ são linguagens amplamente utilizadas no desenvolvimento de bibliotecas nativas, aplicações de alto desempenho e integrações com *SDKs* e bibliotecas externas [13]. Neste trabalho, C++ é utilizado no módulo nativo de rastreamento, responsável por encapsular o acesso ao sensor de profundidade e executar o processamento necessário para identificar peças físicas sobre a mesa.

Esse módulo expõe funções para a aplicação Unity em C#, permitindo que os dados processados, como identificadores e coordenadas das peças, sejam consumidos pelo VTT por meio de interoperabilidade com a biblioteca nativa.

3) OpenCV: OpenCV, sigla para *Open Source Computer Vision Library*, é uma biblioteca de código aberto voltada ao processamento de imagens e aplicações de visão computacional. Bradski descreve a biblioteca como uma infraestrutura para extração e processamento de informações significativas a partir de imagens [14].

4) Kinect for Windows SDK: O Kinect for Windows SDK é o kit de desenvolvimento disponibilizado para integração de aplicações com o sensor Kinect. Ele funciona como uma camada de abstração entre o hardware e o *software*, permitindo acessar os dados capturados pelo dispositivo sem a necessidade de manipular diretamente seus *drivers* [17]. Entre os fluxos disponibilizados pelo SDK estão os dados de cor e de profundidade, que podem ser utilizados por aplicações gerenciadas em C# ou nativas em C++.

5) Interoperabilidade e P/Invoke: Como o protótipo combina uma aplicação Unity em C# com rotinas de processamento nativo em C/C++, utiliza-se interoperabilidade entre código gerenciado e não gerenciado. O *Platform Invocation Services* (P/Invoke) é um mecanismo do .NET que permite ao código gerenciado acessar funções e estruturas expostas por bibliotecas nativas (DLLs), descrevendo a interface e regras de *marshalling* no lado gerenciado [15]. Em C#, esse acesso é normalmente realizado por meio do atributo *DllImport* [16].

D. Arquitetura de Hardware

1) O Sensor Kinect: Lançado em 2010, o Microsoft Kinect popularizou sensores de profundidade de baixo custo, viabilizando captura 3D em tempo real para aplicações interativas [7]. O dispositivo integra um sensor de profundidade, uma câmera RGB e um arranjo de microfones; porém, no contexto deste trabalho, o fluxo de profundidade é o componente central para o rastreamento de peças físicas.

No Kinect v1, a aquisição de profundidade baseia-se em luz estruturada: um projetor infravermelho emite um padrão (*speckle*) que é observado por uma câmera infravermelha. A profundidade é estimada a partir da deformação desse padrão, utilizando triangulação e a geometria conhecida entre projetor e câmera [7]. O resultado é um mapa de profundidade em tempo real, no qual cada pixel armazena uma estimativa de distância em relação ao sensor, servindo como base para o processamento e a inferência de posições no ambiente de mesa.

E. Rastreamento de Objetos

O rastreamento de objetos é responsável por identificar a presença de peças físicas sobre a mesa e manter sua associação com elementos digitais ao longo do tempo. Neste trabalho, esse processo é utilizado para permitir que a movimentação de uma peça real seja refletida em um *token* virtual dentro do VTT.

1) Rastreamento sem Marcadores: o processamento do mapa de profundidade é o cerne do rastreamento de objetos, podendo ser realizado com ou sem marcadores visuais. Em abordagens baseadas em marcadores, os objetos recebem identificadores artificiais, como etiquetas ou padrões visuais, facilitando sua detecção [18]. Já no rastreamento sem marcadores, busca-se reconhecer os objetos a partir de suas características físicas ou da alteração que provocam no ambiente observado.

2) Segmentação por subtração de fundo e limiarização: A segmentação tem como objetivo separar regiões de interesse em uma imagem para permitir processamento posterior [19]. Em sensores de profundidade, uma estratégia comum é comparar o quadro atual com uma referência da mesa vazia, destacando regiões cuja profundidade foi alterada (*background subtraction*). Em seguida, aplica-se limiarização (*thresholding*) para obter uma máscara binária que separa primeiro plano e fundo, reduzindo a influência de ruído do sensor [19].

Em cenários *tabletop* com Kinect, abordagens baseadas em limiar de profundidade por pixel são utilizadas para extrair objetos em primeiro plano [20]. No contexto de RPG com mesa aumentada e projeção, técnicas de profundidade e visão computacional também são empregadas para suportar interação e visualização compartilhada [8].

3) Morfologia Matemática: Após a segmentação, a máscara binária pode apresentar ruído, falhas e fragmentação devido a oscilações do sensor. Para reduzir esses efeitos, aplicam-se operações de morfologia matemática, que transformam regiões com base em sua forma por meio de um elemento estruturante [19].

As operações fundamentais são erosão e dilatação. A erosão tende a remover componentes pequenos e ruídos isolados, enquanto a dilatação expande regiões detectadas, podendo preencher lacunas e reconectar partes próximas. Portanto, neste trabalho, operações morfológicas (por exemplo, abertura e/ou fechamento) são utilizadas para estabilizar a máscara antes da etapa de identificação individual de peças.

4) Agrupamento de Regiões e Descritores Geométricos: Com a máscara refinada, as regiões em primeiro plano podem ser agrupadas a partir da conectividade espacial dos pixels, permitindo separar objetos distintos na cena. Abordagens desse tipo são comuns em aplicações *tabletop* com Kinect para isolar

objetos posicionados sobre uma superfície a partir de dados de profundidade [20].

Após o agrupamento, cada região pode ser descrita por propriedades geométricas, como área, retângulo envolvente e centróide. O centróide é especialmente relevante neste trabalho por representar uma estimativa robusta da posição da peça sobre a mesa, sendo utilizado como entrada para conversão de coordenadas e atualização do *token* correspondente no ambiente virtual.

5) Associação de Dados e Rastreamento Temporal Heurístico: Rastreamento pode ser entendido como a atribuição de rótulos consistentes a objetos ao longo de múltiplos quadros (*frames*) [21]. Entre as categorias de rastreamento, o *point tracking* representa cada objeto por um ponto (por exemplo, centróide) e realiza a correspondência entre quadros com base no estado anterior, assumindo movimento relativamente suave [21].

F. Metodologia de Desenvolvimento

No desenvolvimento de *software*, é comum aplicar metodologias ágeis, que priorizam a entrega incremental, a criação de artefatos mínimos de engenharia e a capacidade de adaptação a mudanças durante o projeto [22]. Para este trabalho, foi adotado o método Kanban, por sua adequação ao desenvolvimento de protótipos e ao acompanhamento contínuo das tarefas.

Segundo Pressman e Maxim [22], o Kanban é uma metodologia enxuta voltada à melhoria de processos e fluxos de trabalho, tendo origem em práticas de engenharia industrial da Toyota e posterior adaptação ao desenvolvimento de *software*. Uma de suas principais práticas é a visualização do fluxo de trabalho por meio de um quadro Kanban, no qual as tarefas avançam entre diferentes estágios de desenvolvimento.

Neste projeto, o quadro foi estruturado nas colunas *A Fazer*, *Em Desenvolvimento*, *Em Teste* e *Concluído*, sendo utilizado para organizar o desenvolvimento progressivo do VTT digital e dos recursos híbridos de rastreamento, projeção e integração com a aplicação Unity.

III. Trabalhos Correlatos

Nesta seção são apresentados trabalhos relevantes para a concepção deste projeto, seguidos de suas respectivas contribuições. Foram selecionados estudos sobre *Virtual Tabletops* (VTTs), que auxiliaram na identificação de funcionalidades essenciais e requisitos recorrentes, e trabalhos sobre interação *tabletop* com sensores de profundidade e projeção, que fundamentaram as decisões relacionadas ao modo híbrido proposto [2], [6], [20].

A. Desenvolvimento de Virtual Tabletops

1) Análise de Jogos de RPG em Mesas Virtuais: Prass [2] apresenta uma análise extensa de aplicações de Mesas Virtuais (Roll20, Fantasy Grounds, MapTool e d20Pro), categorizando suas funcionalidades em cinco grupos: características do sistema, comunicação, tabuleiros e mapas, personagens e miniaturas, e rolagem de dados e automação. O autor propõe o conceito de *Mesa Virtual Ideal*, que integraria os melhores atributos dos RPGs de Mesa (liberdade, improvisação e interação social) com os dos RPGs Eletrônicos (agilidade, visualização e automação de regras). A principal contribuição deste trabalho reside na identificação sistemática dos requisitos funcionais essenciais que um VTT deve oferecer, incluindo sistema agnóstico, suporte para personalização de regras, automação de cálculos, gerenciamento visual de mapas e *tokens*, e integração de comunicação. Esses princípios serviram como base para o design modular do VTT híbrido proposto neste trabalho, especialmente a ênfase na preservação da liberdade criativa aliada à eficiência computacional.

B. Rastreamento de Objetos Físicos

1) Physically Interactive Tabletop Augmented Reality Using the Kinect: Corbett-Davies et al. [20] apresentam um sistema de realidade aumentada para *tabletop* que utiliza Microsoft Kinect posicionado aproximadamente 1 metro acima da superfície para rastrear objetos em 6 graus de liberdade (6DOF). O método emprega segmentação por profundidade para extrair objetos em primeiro plano, agrupamento por componentes conectados e rastreamento temporal do movimento, visando obter estabilidade suficiente para interação em tempo real. Além disso, o artigo discute desafios práticos como ruído e oclusões, comuns em cenários de mesa com manipulação direta. Esse fluxo de segmentação, subtração de fundo, agrupamento e rastreamento temporal fornece a base técnica para detectar e acompanhar a movimentação de peças físicas sobre o tabuleiro digital proposto.

2) TIPTAB: A Tangible Interactive Projection Tabletop for Virtual Experiments: Yuan et al. apresentam o TIPTAB, uma mesa interativa com projeção e objetos tangíveis rastreados por câmera de profundidade, com foco em integrar entrada física e saída digital no mesmo espaço de interação [6]. O artigo descreve um *pipeline* que inclui processamento do mapa de profundidade, detecção de objetos e, principalmente, uma etapa explícita de *coordinate mapping* e *registration* para alinhar o conteúdo projetado ao ambiente físico. Os autores destacam que a consistência entre o rastreamento e a projeção é central para reduzir esforço de interação e manter coerência visual durante o uso. Esse trabalho contribuiu ao fundamentar a necessidade de um mapeamento consistente entre o espaço do sensor e o espaço do tabuleiro/projeção, orientando a abordagem de calibração e integração adotada neste projeto.

IV. Metodologia

O desenvolvimento deste trabalho foi conduzido em duas fases principais: (i) desenvolvimento de um *software* VTT 2D para desktop e (ii) implementação do sistema de rastreamento físico e realidade aumentada projetiva. Para o gerenciamento do projeto, foi adotado o método Kanban, conforme descrito na Seção II-F.

A. Visão Geral do Sistema

A arquitetura do protótipo integra três componentes principais: (i) um VTT 2D desenvolvido em Unity/C#, responsável

por renderização, interface e gerenciamento do estado da sessão; (ii) um sensor de profundidade Microsoft Kinect, utilizado para detectar a presença e estimar a posição de peças sobre a mesa; e (iii) um projetor, utilizado para exibir o mapa e o *feedback* visual sobre a superfície física.

Conforme ilustrado na Figura 1, o Kinect é posicionado acima da superfície de jogo e captura continuamente quadros do fluxo de profundidade. Esses quadros são processados em uma camada nativa (DLL), responsável por segmentar regiões em primeiro plano e estimar posições das peças detectadas. A aplicação Unity consome os resultados por interoperabilidade (P/Invoke), atualiza os *tokens* e demais estados visuais no VTT e produz a saída para monitor e/ou projeção.

O ciclo operacional segue um fluxo contínuo: (1) captura de profundidade pelo Kinect; (2) processamento e estimativa de posições na DLL; (3) envio de identificadores e coordenadas das peças rastreadas para a Unity; e (4) atualização dos *tokens* e renderização da saída. Essa separação permite isolar o processamento mais próximo ao hardware em uma biblioteca nativa, enquanto a Unity atua como camada de apresentação e interação. Com isso, o núcleo do VTT permanece desacoplado das dependências do sensor, preservando a responsividade da interface gráfica e permitindo a evolução incremental do protótipo.

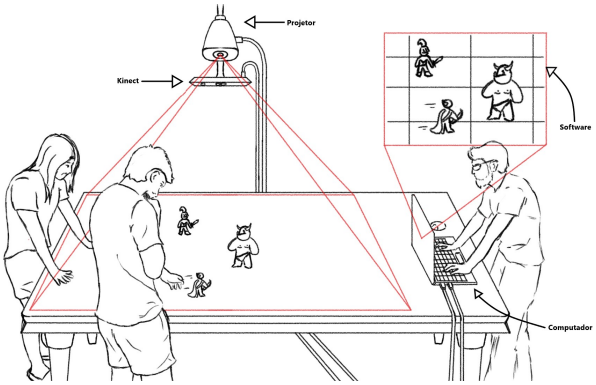


Figura 1: Arquitetura geral do sistema.

B. Levantamento e Definição de Requisitos

As especificações do sistema foram definidas a partir de um levantamento de requisitos, considerando funcionalidades recorrentes em VTTs [2] e necessidades adicionais do modo híbrido (rastreamento físico e projeção). Os requisitos foram revisados quanto à viabilidade técnica e organizados em requisitos funcionais e não funcionais [22].

As metas de desempenho do rastreamento (RNF01–RNF04) foram definidas como critérios operacionais para avaliação do protótipo. Esses critérios foram inspirados em preocupações recorrentes em sistemas tabletop com sensor de profundidade e projeção, como atualização em tempo real, estabilidade espacial, mapeamento entre espaço físico e digital e suporte à manipulação de objetos tangíveis [6], [20]. Assim, os valores adotados na Tabela I devem ser interpretados como metas experimentais do presente trabalho, e não como limites normativos extraídos diretamente da literatura.

Os requisitos foram inseridos em um quadro Kanban, utilizado como *backlog* e registro de progresso. As colunas do quadro foram estruturadas como *A Fazer*, *Em Desenvolvimento*, *Em Teste* e *Concluído* permitindo priorização e acompanhamento das entregas do protótipo.

Tabela I: Requisitos do Sistema (Funcionais e Não Funcionais)

ID	Descrição
RF01	Gerenciar criação de personagens com atributos customizáveis
RF02	Renderizar mapas 2D em um *grid*
RF03	Implementar sistema de *tokens* para representar personagens
RF04	Controlar personagens jogadores e não-jogadores
RF05	Realizar rolagem automatizada de dados
RF06	Implementar sistema de névoa de guerra (*fog of war*) com visibilidade diferenciada para mestre e Jogadores
RF07	Rastrear posição de peças físicas em tempo real
RF08	Detectar movimentação de peças pelo usuário
RF09	Projetar interface visual sobre a superfície
RF10	Projetar *feedback* visual na superfície física
RF11	Sincronizar estado físico-digital
RNF01	Taxa de atualização mínima de 15 FPS
RNF02	Latência de rastreamento inferior a 100ms
RNF03	Precisão de posicionamento de ±5mm
RNF04	Suporte para até 8 peças simultâneas

C. Projeto de software

Diagramas *Unified Modeling Language* (UML) permitem representar diferentes visões de um sistema durante o projeto de *software*. Como o desenvolvimento adotou uma abordagem ágil, com menor ênfase em documentação formal, este trabalho apresenta apenas um modelo conceitual de domínio [22]. Portanto, o diagrama não deve ser interpretado como um diagrama de classes definitivo, mas como uma síntese dos principais elementos do sistema e de suas relações.

O modelo organiza o protótipo a partir da classe `VTT`, que gerencia o modo de operação (*Digital* ou *Híbrido*) e a camada ativa do tabuleiro. O conteúdo é estruturado por `Camada` e `Mapa`, sobre os quais são posicionados `Tokens`, podendo estar associados a um `Personagem`. A visibilidade do cenário é tratada por `NévoaDeGuerra`, enquanto `TelaMestre` e `TelaJogadores` representam perspectivas distintas de visualização.

No modo híbrido, `SistemaRastreamento` e `SensorKinect` representam a etapa de captura e processamento dos dados de profundidade. O sistema realiza calibração, processamento de quadros e detecção de peças físicas, utilizando essas informações para atualizar a posição dos `Tokens` quando o rastreamento está ativo.

D. Desenvolvimento do software VTT

O *software* VTT foi estruturado em controladores especializados, cada um responsável por um conjunto específico de funcionalidades, como carregamento de mapas, persistência local de personagens, manipulação de *tokens*, rolagem de dados, controle da névoa de guerra e comunicação entre módulos por

eventos. Essa organização buscou reduzir o acoplamento entre as partes do sistema e facilitar a manutenção e expansão do protótipo.

A partir dessa estrutura, foram implementadas as funcionalidades essenciais de apoio à condução de sessões de RPG, incluindo o gerenciamento de mapas, personagens, elementos visuais e mecanismos de interação utilizados pelo mestre e pelos jogadores.

1) Sistema de Mapas e Renderização do Grid: O subsistema de mapas permite carregamento dinâmico de cenários sem pré-configuração rígida. A aplicação seleciona imagens (*PNG/JPG*) do armazenamento local, converte-as para `Texture2D` e as renderiza via `SpriteRenderer` como base visual do tabuleiro. Sobre o mapa, o *software* gera procedimentalmente uma grade (*grid*) por meio de `LineRenderer`, permitindo ajustar escala e alinhamento conforme a necessidade da sessão.

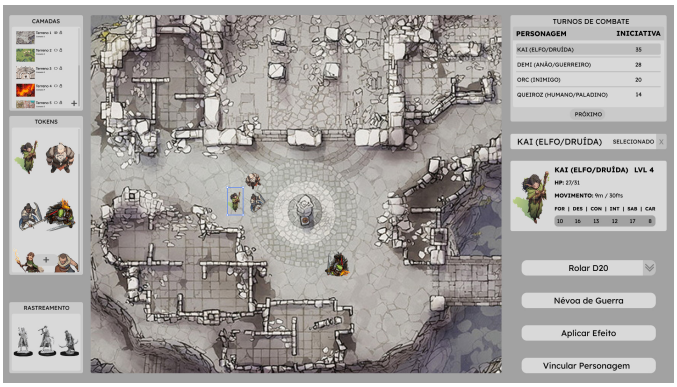
2) Gerenciamento Persistente de Personagens: As entidades narrativas foram abstraídas em estruturas de dados serializáveis (`CharacterRecord`). O gerenciamento dessas informações (`CharacterManager`) implementa salvamento local em arquivos JSON, permitindo armazenamento de atributos customizados em formato chave-valor (*Dictionary*). Para reduzir recarregamentos e evitar alocações repetidas, o *software* utiliza *cache* de retratos (`avatarCache`), mantendo texturas em memória durante a sessão e liberando recursos quando necessário.

3) Sistema de tokens e interação: A representação dos elementos no tabuleiro é realizada por um sistema de *tokens*, responsável por criar, selecionar, mover e atualizar peças virtuais associadas a personagens, inimigos ou objetos de cena. Cada *token* mantém informações de estado, como posição, seleção, visibilidade e, quando aplicável, vínculo com uma peça física rastreada pelo Kinect. O módulo também informa alterações relevantes para outros componentes do VTT, como a Névoa de Guerra.

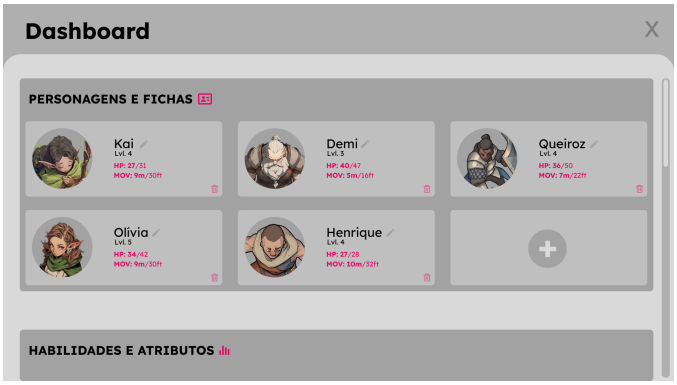
4) Névoa de Guerra e visibilidade: A Névoa de Guerra é implementada como uma camada de oclusão sobreposta ao mapa, atualizada conforme ações do mestre e/ou alcance de visão associado aos *tokens*. A renderização utiliza *shader* para compor o efeito visual no tabuleiro, permitindo ocultar ou revelar áreas do mapa e diferenciar a visualização do mestre daquela exibida aos jogadores.

5) Controle de câmera e sistema de coordenadas: A navegação no tabuleiro permite operações de *pan*, *zoom* e foco no mapa ativo. Para manter consistência entre interação manual, posicionamento de *tokens* e rastreamento físico, o sistema realiza conversões entre coordenadas de mundo e coordenadas normalizadas do mapa ($[0, 1]$). Essa padronização permite posicionar elementos de forma consistente em diferentes mapas e camadas.

6) Interface do mestre e visualização dos jogadores: A interface do mestre concentra as ações de gerenciamento do VTT, incluindo mapas, camadas, *tokens*, névoa de guerra e ferramentas de interação. A visualização dos jogadores é tratada separadamente, utilizando uma câmera dedicada e uma textura de renderização para exibição em segunda tela ou projeção. Essa separação permite que o mestre mantenha controles e informações privadas, enquanto os jogadores visualizam apenas o conteúdo destinado à sessão.



(a) Protótipo inicial da tela do sistema de mapas mostrando grid, camadas de terreno e *tokens*.



(b) Protótipo inicial da tela de gerenciamento de personagens e fichas.

Figura 2: Protótipos das telas do VTT no modo digital.

E. Rastreamento e Processamento das Peças Físicas

O rastreamento foi implementado como um *pipeline* que utiliza o fluxo de profundidade do Kinect para identificar variações na superfície da mesa e estimar a posição de objetos físicos no espaço observado pelo sensor. O processo inicia com uma calibração da mesa vazia, armazenada como mapa de profundidade de referência $D_{bg}(x, y)$. Em seguida, os quadros capturados são processados pelo módulo nativo de rastreamento, com apoio do OpenCV, e convertidos em identificadores e coordenadas das peças físicas, posteriormente consumidos pela aplicação Unity para atualização dos *tokens* virtuais.

Neste projeto, adotou-se uma abordagem sem marcadores para preservar a aparência original das peças utilizadas na mesa. Embora essa escolha possa reduzir a robustez diante de objetos geometricamente semelhantes, ela mantém a integridade estética dos componentes físicos e favorece uma experiência mais próxima do RPG presencial.

1) Segmentação por subtração de fundo e limiarização: Durante a execução, seja $D_t(x, y)$ o mapa de profundidade

no instante t. A segmentação baseia-se na diferença absoluta em relação ao fundo:

$$\Delta D_t(x, y) = |D_t(x, y) - D_{bg}(x, y)| . \tag{1}$$

A partir dessa diferença, obtém-se uma máscara binária $M_t(x, y)$ por limiarização em faixa:

$$M_t(x, y) = \begin{cases} 1, & T_{min} \leq \Delta D_t(x, y) \leq T_{max} \\ 0, & \text{caso contrário.} \end{cases} \tag{2}$$

Os limiares foram definidos empiricamente para destacar alterações compatíveis com as peças físicas sobre a mesa, gerando uma máscara binária de regiões candidatas. Essa máscara serve como entrada para as etapas posteriores de refinamento, agrupamento e extração de posição.

2) Refinamento morfológico e extração de regiões: Para reduzir ruído e fragmentação, aplica-se filtragem mediana e erosão morfológica com elemento estruturante B:

$$M'_t = \text{median}(M_t) \ominus B. \tag{3}$$

Em seguida, a máscara é refinada e as regiões em primeiro plano são extraídas por contornos externos. Cada região candidata é avaliada por seu centróide e por propriedades geométricas simples, como área, proporção e circularidade, permitindo descartar ruídos, oclusões e artefatos do cenário [19].

3) Associação heurística temporal: Para manter a identidade das peças ao longo do tempo, realiza-se associação temporal por proximidade entre centróides detectados em quadros consecutivos [21]. Dada a posição predita de uma peça $\hat{\mathbf{p}}_i(t)$ e um centróide detectado $\mathbf{q}_j(t)$, utiliza-se a distância Euclidiana:

$$d_{ij} = \|\hat{\mathbf{p}}_i(t) - \mathbf{q}_j(t)\|_2 . \tag{4}$$

Neste projeto, cada peça física é representada por seu centróide, e a associação entre quadros consecutivos é realizada por proximidade espacial. Para isso, calcula-se a distância Euclidiana entre as detecções atuais e as posições previamente estimadas, selecionando as correspondências de menor distância dentro de um limiar máximo de deslocamento.

Para lidar com oclusões e falhas momentâneas de detecção, aplica-se uma tolerância temporal baseada em contadores: quando uma peça não é observada em determinado quadro, sua última posição válida é mantida por uma janela limitada antes que seu identificador seja considerado perdido. Assim, o protótipo adota um rastreamento temporal heurístico por centróide, fundamentado nos princípios gerais de rastreamento por pontos descritos por Yilmaz et al. [21].

4) Mapeamento para o sistema de coordenadas do VTT: As posições estimadas pelo rastreador são convertidas para o espaço do tabuleiro por meio de uma região de interesse (*Region of interest*) correspondente à área jogável. No protótipo, a região de interesse é definida a partir de quatro pontos de referência observados pelo Kinect. A partir dessas correspondências, calcula-se uma homografia $\mathbf{H}$, que mapeia coordenadas em pixel (x, y) para coordenadas normalizadas no plano do tabuleiro $(x_n, y_n) \in [0, 1]$ [6]:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \mathbf{H} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad x_n = \frac{u}{w}, \quad y_n = \frac{v}{w}. \tag{5}$$

Por fim, as coordenadas normalizadas são convertidas para o espaço da Unity por interpolação linear entre os limites do mapa ativo, considerando os ajustes finos de calibração. Como as coordenadas da imagem têm origem no canto superior esquerdo, a conversão ajusta o eixo vertical para manter a correspondência correta com o sistema de coordenadas do VTT. Essa abordagem é suficiente para o cenário de mesa com vista superior e permite integrar o rastreamento físico diretamente aos *tokens* virtuais.

F. Integração do rastreamento ao software VTT

A integração entre a aplicação Unity (C#) e a camada nativa de rastreamento é realizada por meio de P/Invoke. A cada quadro, o VTT executa a rotina nativa, obtém as peças detectadas e converte suas coordenadas do espaço do Kinect para o espaço normalizado da projeção, utilizando a homografia definida na calibração.

Após a conversão, o sistema descarta peças perdidas, coordenadas inválidas ou posições fora da região de interesse. As coordenadas restantes são limitadas ao intervalo $[0, 1]$, convertidas para o espaço do mapa ativo na Unity e usadas para atualizar o *token* associado à peça física. Essa etapa mantém a sincronização entre a movimentação real das peças e sua representação visual no tabuleiro digital. O Listing 1 resume esse fluxo.

Listing 1: Pseudocódigo da integração Kinect–VTT

```
Para cada quadro:
  se a calibracao nao for valida:
    encerrar atualizacao

  executar processamento nativo do Kinect
  obter quantidade de pecas detectadas
  obter limites do mapa ativo

  para cada peca detectada:
    obter id, posicao em pixel e estado

    se a peca estiver perdida:
      continuar

    uv <- converter pixel do Kinect para projecao

    se uv for invalido ou estiver fora da regiao de
        interesse:
      continuar

    uv <- limitar uv ao intervalo [0,1]
    posicaoMapa <- converter uv para espaco do mapa
    posicaoMapa <- aplicar ajustes finos

    se existir token associado ao id:
      atualizar posicao do token
```

O Listing 1 abstrai a rotina de integração implementada no sistema. A chamada nativa atualiza os dados de rastreamento, enquanto a homografia realiza o mapeamento entre Kinect e projeção. A validação pela região de interesse evita que

detecções externas à área jogável interfiram no tabuleiro, e a etapa final mantém a correspondência entre peças físicas e *tokens* virtuais.

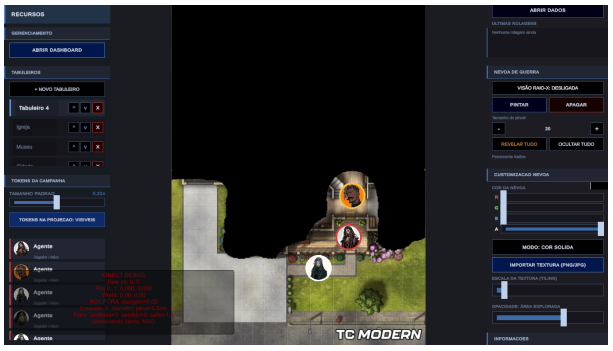
V. Resultados

Esta seção apresenta os resultados obtidos com o protótipo desenvolvido, considerando as funcionalidades implementadas no VTT, a integração híbrida com projeção e rastreamento físico, as medições de desempenho e a avaliação exploratória em uma sessão presencial de RPG.

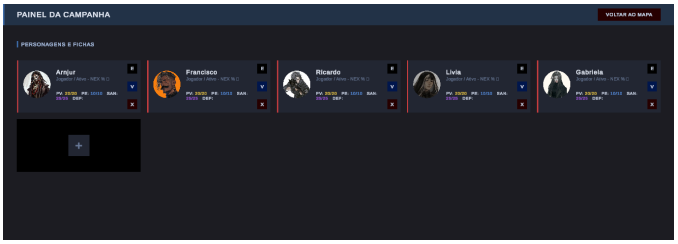
A. Funcionalidades implementadas

O protótipo opera em *desktop* e permite carregar mapas 2D, organizar elementos em camadas, posicionar *tokens*, gerenciar personagens, realizar rolagens de dados e controlar a névoa de guerra. A Figura 3 apresenta as principais telas do modo digital: (a) tabuleiro com mapa, *grid* e *tokens*; (b) controle de visibilidade por névoa de guerra; e (c) gerenciamento de personagens e fichas.

(a) Tabuleiro com mapa, *grid* e *tokens*.



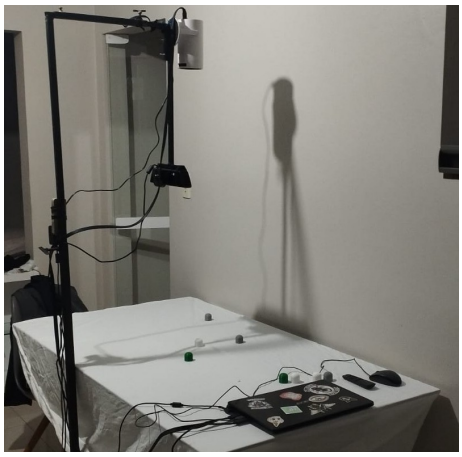
(b) Controle de visibilidade por névoa de guerra.



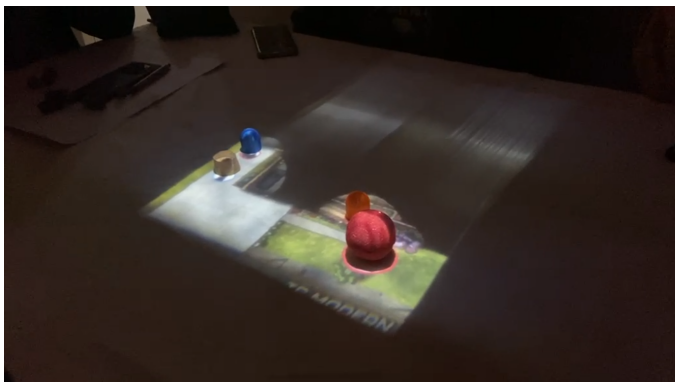
(c) Gerenciamento de personagens e fichas.

Figura 3: Telas do VTT no modo digital.

No modo híbrido, o mapa digital é projetado sobre a mesa e os *tokens* virtuais são atualizados a partir do rastreamento das peças físicas. Dessa forma, o sistema mantém a manipulação presencial das peças e a atenção compartilhada sobre a superfície de jogo, ao mesmo tempo em que incorpora recursos digitais típicos de VTTs. A Figura 4 apresenta o ambiente experimental, com Kinect e projetor posicionados acima da mesa, bem como uma sessão presencial em execução.



(a) Estrutura física do protótipo.



(b) Sessão com projeção ativa e peças físicas.

Figura 4: Contexto de uso do protótipo no modo híbrido.

A Figura 5 exemplifica o rastreamento das peças físicas e sua correspondência com os *tokens* no tabuleiro, incluindo a região de interesse usada no mapeamento entre o espaço físico e o espaço virtual. Além das imagens, foram produzidos vídeos demonstrativos do protótipo em funcionamento, registrando a projeção do mapa, a movimentação das peças físicas e a atualização dos *tokens* durante a execução. Os vídeos estão disponíveis em: https://youtube.com/shorts/hwpM_eLdsM4?feature=share e https://youtube.com/shorts/ssJBOcfAQyc?feature=share.

B. Desempenho e rastreamento

O desempenho foi avaliado em cenário de uso, considerando tanto a atualização visual do VTT quanto o rastreamento das peças físicas no modo híbrido. O protótipo foi instrumentado em tempo de execução por meio de um módulo de diagnóstico responsável por calcular a taxa de quadros da aplicação, o tempo médio por quadro e registrar amostras em arquivo CSV.

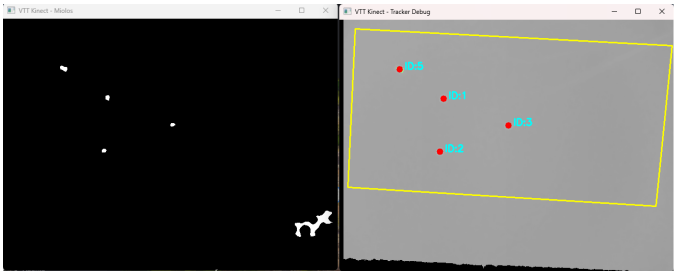


Figura 5: Peças físicas rastreadas e atualização de *tokens* no tabuleiro.

A latência do rastreamento corresponde ao tempo gasto na chamada da rotina nativa `ProcessFrame`, antes da atualização dos dados consumidos pela Unity. Também foram registrados a frequência estimada do rastreamento e a quantidade de peças detectadas no último quadro.

A precisão do posicionamento foi estimada pela comparação entre pontos físicos de referência na superfície projetada e suas posições correspondentes no espaço virtual após a calibração, não constituindo uma medição metrológica formal. A quantidade de peças rastreadas simultaneamente foi avaliada por testes progressivos com múltiplos objetos dentro da região de interesse. A Tabela II compara as metas definidas nos requisitos não funcionais com os valores observados.

Tabela II: Métricas de desempenho: metas dos RNFs e valores observados

Métrica	**Meta**	**Observado**
Taxa de atualização do VTT	$\geq$ 15 FPS	$\approx$ 30 FPS
Latência do rastreamento	$\leq$ 100 ms	$\approx$ 30 ms
Precisão de posicionamento	$\pm$ 5 mm	erro estimado $<$ 3 mm
Peças simultâneas rastreadas	$\geq$ 8	10 peças testadas

Os resultados indicam que o protótipo atendeu às metas não funcionais definidas para o modo híbrido no cenário experimental avaliado. A taxa de atualização permaneceu acima do mínimo estabelecido, permitindo visualização contínua do tabuleiro e dos *tokens*. A latência aproximada de 30 ms ficou abaixo do limite de 100 ms, contribuindo para que a movimentação das peças físicas fosse refletida de forma responsiva no ambiente virtual. O erro estimado inferior a 3 mm, obtido após calibração em pontos de referência da superfície projetada, ficou abaixo da margem definida de $\pm$ 5 mm, indicando alinhamento adequado no cenário testado.

Quanto à robustez, o rastreamento manteve comportamento estável em condições usuais de uso, especialmente quando a calibração estava correta e as peças permaneciam dentro da região de interesse. Em movimento lento ou moderado, o acompanhamento das peças foi consistente. No entanto, foram observadas instabilidades em situações de oclusão pelas mãos, proximidade excessiva entre peças, movimentação simultânea de múltiplos objetos e posicionamento fora da área calibrada. Embora o sistema possa suportar maior quantidade de objetos em condições controladas, este trabalho considera apenas as nove peças efetivamente testadas como evidência experimental.

C. Avaliação exploratória e limitações

Foi realizada uma avaliação exploratória durante uma sessão presencial de RPG com duração aproximada de quatro horas, utilizando o protótipo em modo híbrido. A sessão contou com três participantes com experiência prévia em RPG, todos atuando como jogadores, enquanto o autor conduziu a partida como mestre. Durante a sessão, foram exercitadas as principais funcionalidades do VTT, incluindo visualização do mapa, manipulação de *tokens*, uso da névoa de guerra, rolagem de dados, gerenciamento de personagens e rastreamento de peças físicas.

Ao final da sessão, foi realizada uma conversa semiestruturada com os participantes, conduzida pelo autor. Os relatos foram registrados por meio de anotações durante e após a sessão, contemplando a visibilidade do tabuleiro, a integração entre peças físicas e digitais, o uso da névoa de guerra e as limitações percebidas. Assim, os dados possuem caráter qualitativo e exploratório.

Os relatos indicaram percepção positiva quanto à visibilidade do tabuleiro, à organização espacial da partida e à integração entre elementos físicos e digitais. Também foi apontado que o sistema preserva características do RPG presencial, como a manipulação física das peças e a interação em torno da mesa, ao mesmo tempo em que incorpora recursos de automação e visualização típicos de VTTs. A revelação dinâmica de áreas do mapa associada ao deslocamento das peças foi mencionada como um recurso relevante para aumentar a sensação de suspense e imersão.

Como limitações, foram observadas falhas eventuais de rastreamento em situações com muitas pessoas interagindo simultaneamente, possíveis interferências relacionadas ao tamanho das peças e a ausência de *tokens* puramente digitais para entidades sem peça física correspondente. Também foram sugeridas melhorias para a névoa de guerra, como revelação por áreas específicas, visão em cone, delimitação de paredes e mecanismos para evitar a revelação indevida de regiões do mapa. Outro ponto sugerido foi a inclusão de *feedback* visual para indicar o *token* ativo durante combates ou cenas organizadas por iniciativa.

Assim, embora os resultados indiquem boa aceitação inicial do protótipo, eles devem ser interpretados como evidências qualitativas e exploratórias, uma vez que a avaliação foi realizada com número reduzido de participantes e em um contexto específico de uso.

VI. Conclusão

Este trabalho apresentou o desenvolvimento de um protótipo de *Virtual Tabletop* (VTT) híbrido para apoio a sessões de RPG de mesa, integrando recursos digitais de visualização e automação com a interação física característica das partidas presenciais. O sistema foi implementado em Unity/C#,

contemplando funcionalidades como carregamento de mapas, organização em camadas, manipulação de *tokens*, gerenciamento de personagens, rolagem de dados, névoa de guerra e separação entre a visualização do mestre e dos jogadores. Como extensão ao modo presencial, foi integrada uma camada de rastreamento com Kinect, processada em uma biblioteca nativa acessada por P/Invoke, permitindo associar peças físicas a *tokens* virtuais e projetar o estado do tabuleiro sobre a mesa.

Os resultados indicam que o protótipo atendeu às metas não funcionais definidas para o modo híbrido no cenário experimental avaliado, alcançando aproximadamente 30 FPS no VTT, latência de rastreamento próxima de 30 ms, erro estimado inferior a 3 mm após calibração em pontos de referência e rastreamento de dez peças simultâneas. A avaliação exploratória, realizada em uma sessão presencial de RPG com três jogadores experientes e duração aproximada de quatro horas, também indicou percepções positivas observadas, com relatos de melhora percebida na visualização do cenário, na organização espacial da partida e na imersão, especialmente pelo uso da névoa de guerra associada ao movimento das peças. Ainda assim, por se tratar de uma avaliação qualitativa com número reduzido de participantes, os resultados devem ser interpretados como indícios iniciais de viabilidade e potencial de uso, e não como validação conclusiva da experiência de jogo.

Como limitações, observaram-se dependência de calibração adequada, sensibilidade a oclusões, proximidade excessiva entre peças e possíveis interferências do ambiente físico. Como trabalhos futuros, propõe-se aprimorar a calibração e a robustez do rastreamento, ampliar os modos de interação com a névoa de guerra, incluir suporte a *tokens* puramente digitais e adicionar recursos de apoio a combate, como indicação visual do *token* ativo. Também se recomenda realizar avaliações com mais participantes e diferentes cenários de jogo para investigar de forma mais ampla o impacto da mesa híbrida na experiência de jogadores e mestres.

Referências

[1] M. Montola e J. Stenros, Eds. *Beyond Role and Play: Tools, Toys and Theory for Harnessing the Imagination*. Ropecon ry, 2004. ISBN: 952-91-6842-X.

[2] H. L. T. da Silva Prass. "Análise de jogos de RPG em mesas virtuais". Trabalho de Conclusão de Curso, Bacharelado em Ciência da Computação, Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brasil, 2017.

[3] M. S. Heck. "Desenvolvimento modular e personalização da experiência do usuário em engenharia de software: um estudo de caso com role-playing games e virtual tabletops". Monografia, Bacharelado em Engenharia de Computação, Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brasil, 2025.

[4] T. Farkas, N. G. J. Hughes e R. Fiebrink. "How boardgame players imagine interacting with technology". Em: *Proceedings of the ACM on Human-Computer Interaction* 8.CHI PLAY (2024), art. 313. DOI: 10.1145/3677078.

[5] R. T. Azuma. "A survey of augmented reality". Em: *Presence: Teleoperators and Virtual Environments* 6.4 (1997), pp. 355–385.

[6] Q. Yuan et al. "TIPTAB: A tangible interactive projection tabletop for virtual experiments". Em: *Computer Applications in Engineering Education* 30.5 (2022), pp. 1350–1369. DOI: 10.1002/cae.22524.

[7] Z. Zhang. "Microsoft Kinect sensor and its effect". Em: *IEEE MultiMedia* 19.2 (2012), pp. 4–10. DOI: 10.1109/MMUL.2012.24.

[8] A. K. Miyazaki, G. P. Rey e M. Marengoni. "Visão computacional e realidade aumentada aplicadas a jogos de RPG de mesa". Em: *XI Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*. Brasília, Brasil, 2012, pp. 124–131.

[9] Unity Technologies. Components. Disponível em: https://docs.unity3d.com/Manual/Components.html. Acessado em: junho de 2026.

[10] Unity Technologies. SpriteRenderer. Disponível em: https://docs.unity3d.com/ScriptReference/SpriteRenderer.html. Acessado em: junho de 2026.

[11] Unity Technologies. LineRenderer. Disponível em: https://docs.unity3d.com/ScriptReference/LineRenderer.html. Acessado em: junho de 2026.

[12] A. Hejlsberg, M. Torgersen, S. Wiltamuth e P. Golde. *The C# Programming Language*. 4. ed. Addison-Wesley, 2011. ISBN: 978-0-321-74176-9.

[13] B. Stroustrup. *The C++ Programming Language*. 4. ed. Addison-Wesley, 2013. ISBN: 978-0-321-56384-2.

[14] G. Bradski. "The OpenCV Library". Em: *Dr. Dobb's Journal* (2000). Disponível em: https://www.drdobbs.com/open-source/the-opencv-library/184404319. Acessado em: junho de 2026.

[15] Microsoft. Platform Invoke (P/Invoke). Disponível em: https://learn.microsoft.com/en-us/dotnet/standard/native-interop/pinvoke. Acessado em: junho de 2026.

[16] Microsoft. DllImportAttribute Class. Disponível em: https://learn.microsoft.com/en-us/dotnet/api/system.runtime.interopservices.dllimportattribute. Acessado em: junho de 2026.

[17] A. Jana. *Kinect for Windows SDK Programming Guide*. Packt Publishing, 2012. ISBN: 978-1-84969-238-0.

[18] T. A. Syed et al. "In-Depth Review of Augmented Reality: Tracking Technologies, Development Tools, AR Displays, Collaborative AR, and Security Concerns". Em: *Sensors* 23.1 (2023), art. 146. DOI: 10.3390/s23010146.

[19] R. C. Gonzalez e R. E. Woods. *Digital Image Processing*. 4. ed. Pearson, 2018. ISBN: 978-1-292-22304-9.

[20] S. Corbett-Davies, R. Green e A. Clark. "Physically interactive tabletop augmented reality using the Kinect". Em: *27th Conference on Image and Vision Computing New Zealand (IVCNZ)*. 2012, pp. 210–215. DOI: 10.1145/2425836.2425880.

[21] A. Yilmaz, O. Javed e M. Shah. "Object tracking: A survey". Em: *ACM Computing Surveys* 38.4 (2006), art. 13. DOI: 10.1145/1177352.1177355.

[22] R. S. Pressman e B. R. Maxim. *Engenharia de Software: Uma Abordagem Profissional*. 9. ed. AMGH, 2021. ISBN: 978-65-5804-011-8.